便是要從以桌上電腦或伺服器為基礎的資料分析與處理作業、過渡到網頁介面，並且設法讓這段崎嶇路途走得較為平順，讓你能夠在付出汗水之後，享受網站平台的甜美果實。

但本書最具野心的目標是說服讀者，即便需要採用兩種強大的程式語言，以之建構超炫超棒的視覺化網站介面，也將會非常有趣，絕對會讓你沉迷其中。

我認為很多可能從事資料視覺化的程式設計師，大都以為**網站開發**和他們想要做的事情（也就是以 Python 和 JavaScript 撰寫程式）之間，存在著鴻溝。對他們來說，網站開發包含一卡車神祕難懂的知識，諸如標示語言、樣式表、腳本程式、以及維護管理，若不使用擁有怪異名稱的工具，如 *Gulp* 或 *Yeoman*，就沒辦法完成。我將會在本書內文裡證明，往日的巨大鴻溝，已經縮減成能輕易跨越的細小河流，你將能夠把心力集中在真正的事務上頭：花費些許心力便能撰寫程式得到需要的功能，把資料交給網站伺服器呈現給外界，請見圖 P-1。

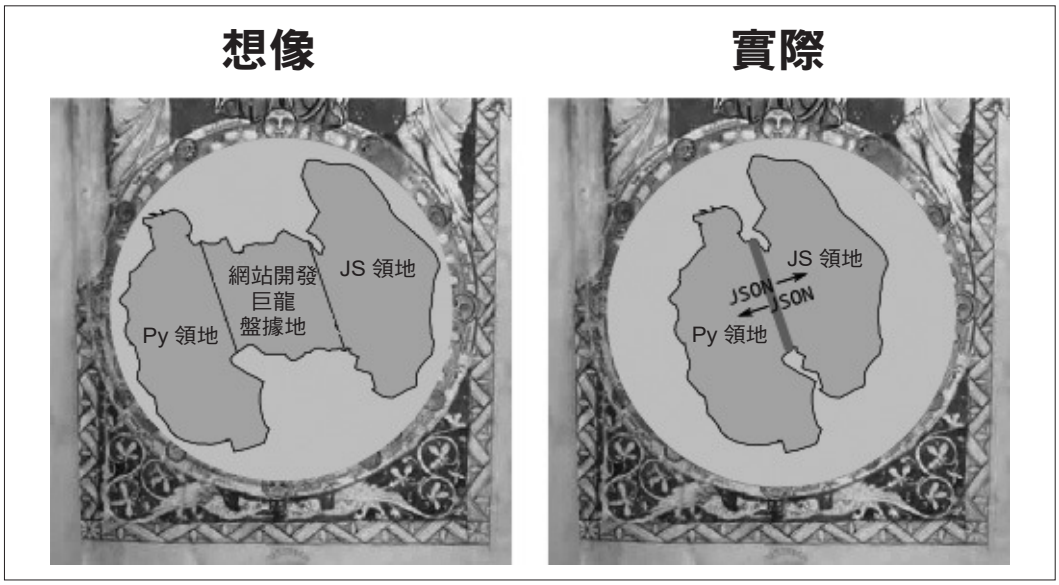


圖 P-1　網站開發巨龍盤據地

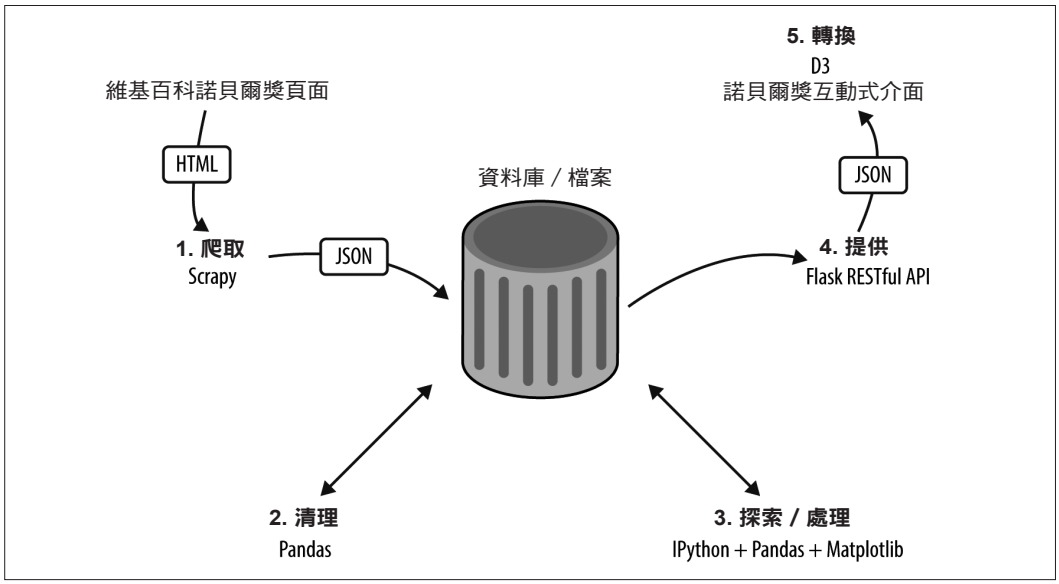


圖 P-2　資料視覺化工具鏈

1. 以 Scrapy 爬取資料

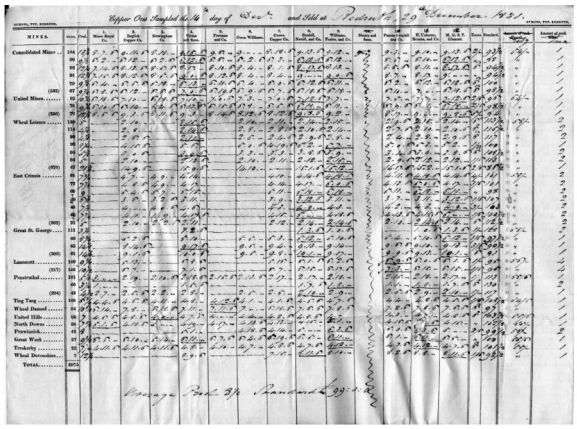
任何資料視覺化任務，首先要面對的難題是設法拿到所需資料，不論是他人的要求或是你自己的個人興趣，若幸運值夠高，將可拿到原始形式，不過一般情況是自己必須出外尋找。使用 Python 從網站（例如網站 API 或 Google 試算表）取得資料的方式有好幾種，我將會一一介紹。諾貝爾獎範例專案需要的資料，將會使用 Scrapy[2] 從維基百科網頁爬取。

Python 程式庫 Scrapy 是一套稱得上工業強度等級的網站爬取器，關於資料流動調節與媒體傳遞途徑，都能輕鬆控制，若你打算爬取相當大量的資料，絕對是不可或缺的工具。想取得感興趣的資料時，爬取網站往往是唯一的途徑，一旦能夠精熟 Scrapy 的作業流程，所有原先遠在天邊的資料，對你來說將只相距一隻蜘蛛 [3]。

2　網站資料爬取（*https://en.wikipedia.org/wiki/Web_scraping*）是種電腦軟體技術，從網站抽取出資訊，通常包含取得網頁與解析的工作。

3　Scrapy 的控制器叫做蜘蛛（spider）。

a.



b.

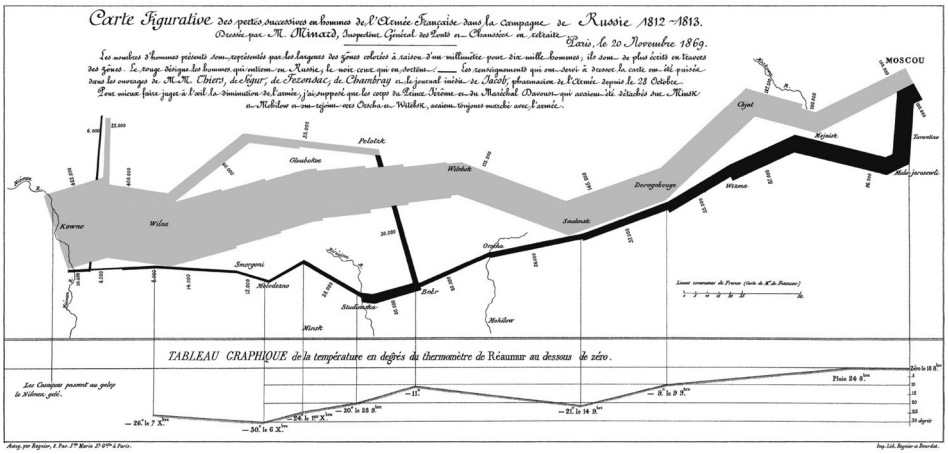


圖 P-3　拿破崙 1812 年征俄戰爭的 (a) 原始試算表與 (b) Joseph Minard 進行視覺化後的模樣

直到最近不久，我們大部分人對於圖表的感知經驗，跟 Charles Minard 面對的觀眾並無太大區別，那些圖表都是事先繪製完成，並且只反映出資料的一個面向，試圖能夠呈現出重要且有意義的那個面向，但不論如何，控制權完全操之在作者手上。從這個角度來看，在傳播資料圖表時，我們只是把真實的墨汁繪點換成電腦螢幕像素罷了。

JavaScript

```
var x = false;
var y = true;
var l = []

if(!x && y === x){...

if(l.length === 0){...
```

Python

```
x = False
y = True
l = []

if not x and y == x:

if l: ...
```

圖 2-3　布林

JavaScript

駝峰式大小寫和底線

```
var studentData = [
    {'name': 'Bob',
     'scores':[68, 75, 56, 81]},
    {name: 'Alice',
     'scores':[75, 90, 64, 88]},
...];
```

無名函式

```
studentData.forEach(function(sdata){
    var av = sdata.scores
        .reduce(function(prev, current){
                    return prev+current;
        },0) / sdata.scores.length;
    sdata.average = av;
```

頭等函數式方法

```
console.log(sdata.name + " scored " +
  sdata.average);


while(i < 10){
...
}

do {
...
}
while(i < 10);
```

Python

```
student_data = [
    {'name': 'Bob',
     'scores':[68, 75, 56, 81]},
    {'name': 'Alice',
     'scores':[75, 90, 64, 88]},
...]
```

斷行

```
s_data = student_data
for data in s_data.items():
    av = sum(data['scores'])\
        /float(len(data['scores']))
    sdata['average'] = av
```

```
print("%s scored %d"%
    (sdata.name, sdata.average));


while i < 10:
    ...

while True:
    if i >= 10:
        break
```

圖 2-4　迴圈和迭代

MongoDB 資料庫預設會在本地端主機執行，預設連接埠號是 27017，但也可放在任何網站上，建立時，也可傳入選用性的使用者名稱和密碼。範例 3-5 示範如何撰寫一支簡單的工具函式，以標準預設值來存取資料庫。

範例 3-5　存取 MongoDB 資料庫

```
from pymongo import MongoClient

def get_mongo_database(db_name, host='localhost',\
                       port=27017, username=None, password=None):
    """ 從 MongoDB 取得某名稱的資料庫，帳密為選用性 """
    # 建立 Mongo 連線
    if username and password:
        mongo_uri = 'mongodb://%s:%s@%s/%s'%\  ❶
        (username, password, host, db_name)
        conn = MongoClient(mongo_uri)
    else:
        conn = MongoClient(host, port)

    return conn[db_name]
```

❶ 在 MongoDB 的 URI（Uniform Resource Identifier，統一資源識別元）裡指定資料庫名稱，因為使用者可能有、可能沒有該資料庫的特殊存取權限。

接下來，可開始建立諾貝爾獎資料庫，並且把得主資料（範例 3-1）放進去。首先取得得主群集，使用字串常數進行存取動作：

```
db = get_mongo_database(DB_NOBEL_PRIZE)
coll = db[COLL_WINNERS]
```

接著要插入得主資料，嘿，怎麼這麼簡單啊，如下：

```
coll.insert(nobel_winners)
輸出：
[ObjectId('55f8326f26a7112e547879d4'),
 ObjectId('55f8326f26a7112e547879d5'),
 ObjectId('55f8326f26a7112e547879d6')]
```

之後想要讀取時，可使用結果陣列裡顯示的 `ObjectId`，MongoDB 已經把它的戳記烙印在我們的 `nobel_winners` 串列裡，加入一個隱藏的 `id` 屬性[8]。

```
nobel_winners
輸出：
[{'_id': ObjectId('55f8326f26a7112e547879d4'),
```

8　MongoDB 很酷的特色之一：ObjectId 是在客户端產生的，因此我們不需要向資料庫查詢。

第四章

網站開發入門

本章要介紹網站開發的核心知識，明白你想爬取資料的網頁結構，了解如何撰寫網頁骨架，作為提供資料視覺化網站與 JavaScript 程式的平台。你將會發現，在現代的網站開發工作中，即便是只擁有一點點基本知識，也會非常有用，特別是你的焦點在於打造自給自足的資料視覺化專案、而非建構整個網站（詳情請見第 84 頁「單一頁面應用程式」）。

一直以來已說過很多次的提醒，也適用於此：此章內容部分屬於參考資料、部分屬於入門指引，裡頭一定有些部分你已熟悉，請自行判斷並跳過，只需研讀新東西即可。

概觀

人類從浩瀚無涯的網際網路汲取資訊，其中佔了大半塊的 WWW（World Wide Web，全球資訊網），基本建構區塊乃是網頁／網站，而網頁又是由數種類型的檔案組成；除了多媒體檔案（圖片、影片、聲音等），網頁的關鍵成分由文字形式的檔案拼湊而成，包括 HTML（HyperText Markup Language，超文件標示語言）、CSS（Cascading Style Sheets，串接樣式表）、JavaScript 腳本程式。此三者，加上其他必需的資料檔案，透過 HTTP（HyperText Transfer Protocol，超文本傳輸協定）傳輸，建構出你眼前的網頁，與你的瀏覽器視窗互動，而網頁的架構又是以 DOM（Document Object Model，文件物件模型）描述，以階層式樹狀結構描述網頁內容。若想要建構新一代的資料視覺化網站，上述種種不可不學，至少需要基本的認知，而此章目標就是迅速帶領各位讀者入門。

圖 4-5 秀出 CSS 如何套用樣式到 HTML 元素，首先選擇元素，以井字號「#」指定 id，以句點「.」指定類別成員。然後定義一或多組屬性 / 值配對，注意屬性 font-family 可以按照優先順序指定一連串的字型，若瀏覽器找不到前面的字型，就會依序使用後面的字型。瀏覽器原本把 font-family 預設為 serif（筆劃有襯線），此處例子希望換成更現代的字型 sans-serif，以 Helvetica Neue 作為第一優先。

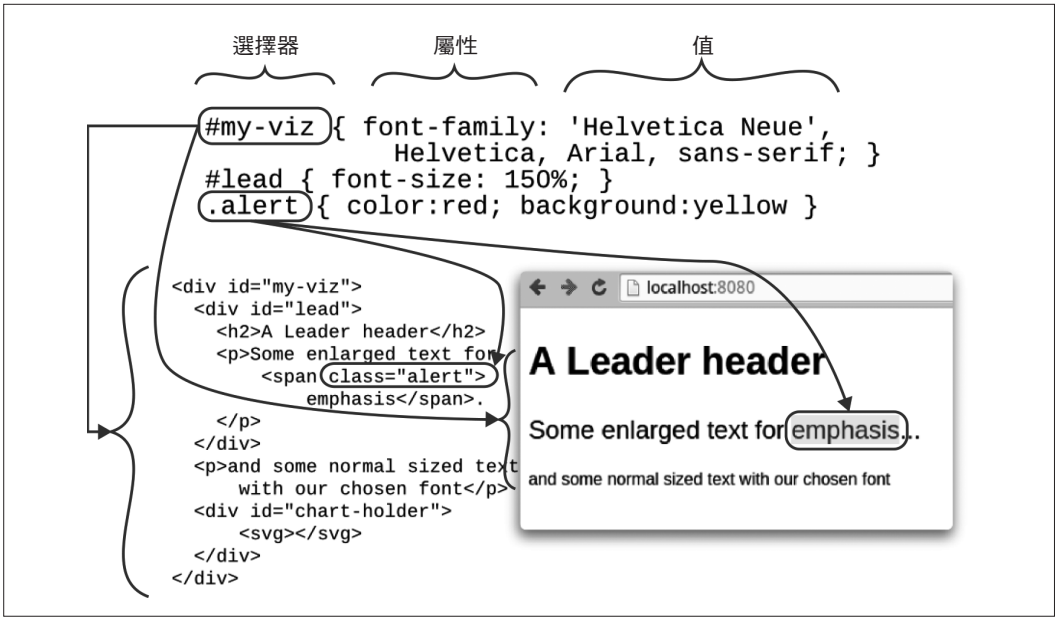


圖 4-5　以 CSS 指定頁面的樣式

CSS 的優先順序規則非常重要，弄明白後才能成功地套用樣式。簡言之，其順序如下：

1. CSS 屬性後若跟著 !important，勝過其他一切。
2. 越特殊的，順序越高（換言之，id 會蓋過 class）。
3. 宣告順序：最後宣告的勝出，但仍敵不過前面的 *1* 與 *2*。

舉個例子，假設有個 ，其 class 為 alert：

```
<span class="alert" id="special-alert">
something to be alerted to</span>
```

元素分頁

若想使用元素分頁，請選 Chrome 右上角選單的更多工具→開發人員工具，或是按下快捷鍵 Ctrl-Shift-I。

圖 4-6 秀出元素分頁的模樣。使用左手邊的放大鏡圖示，便可選擇頁面裡的 DOM 元素，然後在右手邊面板裡觀看它的 HTML 分支。右方面板允許你觀看套用在該元素的 CSS 樣式，並且檢查附加的任何事件聆聽者（event listener）或 DOM 屬性。

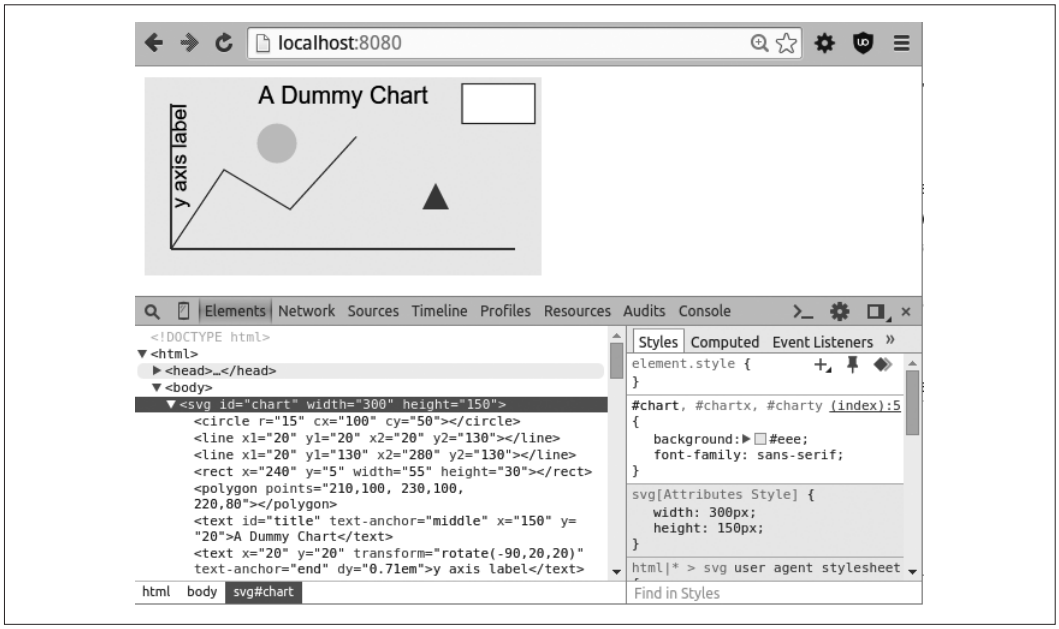


圖 4-6　Chrome 開發人員工具的元素分頁

元素分頁有個非常酷的功能，讓你能以互動方式改變元素的樣式，包括 CSS 樣式與屬性項[8]。憑藉此功能，我們可逐步修改樣式，改善資料視覺化網站的外觀外貌。

Chrome 的元素分頁非常棒，讓我們可以輕易地探索頁面的結構，了解各元素的位置如何被擺放。以 `position` 與 `float` 屬性來排版內容區塊，搞懂其規則，看看套用 CSS 樣式帶來的好處，真的很不錯，請試著提昇你的能力並學習一些有用的技巧。

8　操控屬性項的能力非常有用，特別是在嘗試讓 SVG（可縮放向量圖形）動起來的時候。

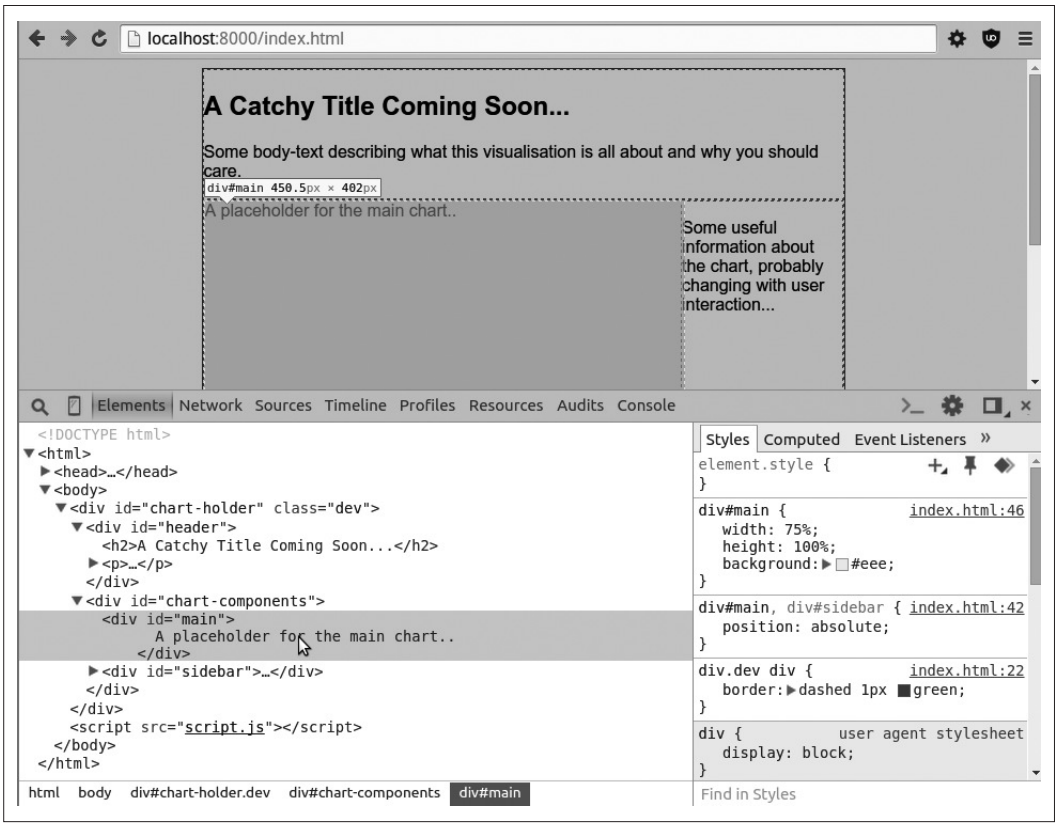


圖 4-8　建構基本網頁

填入內容（血肉）

以 HTML 定義內容區塊、以 CSS 指定位置，接下來，現代的資料視覺化網站專案會使用 JavaScript 來產生出互動式圖表、選單、表格等等元件。若要在近代瀏覽器裡建立視覺元件（除了圖像標籤或多媒體標籤），作法有很多種，主要方式條列如下：

- SVG（Scalable Vector Graphics，可縮放向量圖形），使用特殊的 HTML 標籤。
- 2D 的 canvas 繪製區域。
- 3D 的 canvas WebGL 繪製區域，允許使用 OpenGL 指令子集。
- 使用新穎的 CSS 樣式，建立出動畫、基本幾何圖形等等。

```
  "date_of_birth": "8 October 1927",
  "date_of_death": "24 March 2002",
  "gender": "male",
  "link": "http:\/\/en.wikipedia.org\/wiki\/C%C3%A9sar_Milstein",
  "name": "C\u00e9sar Milstein",
  "place_of_birth": "Bah\u00eda Blanca ,  Argentina",
  "place_of_death": "Cambridge , England",
  "text": "C\u00e9sar Milstein , Physiology or Medicine, 1984",
  "year": 1984
}
```

除此之外，此章還準備爬取得主的照片（若有的話）、以及一些生平履歷資訊（見圖 6-1）。之後將使用圖片（得主照片）與文字（得主的傳記），為我們的諾貝爾獎資料視覺化網站範例，增加一點個性。

圖 6-1　諾貝爾獎得主頁面的爬取目標

設定 Scrapy

Scrapy 應是 Anaconda 整合套件的一份子（見第 1 章），所以你應該已經擁有，若無，可使用底下的 conda 指令進行安裝：

```
$ conda install -c https://conda.anaconda.org/anaconda scrapy
```

```
ax.set_xticks(range(12))  ❷
ax.set_yticks(ticks=range(len(foo_data)))
ax.set_yticklabels(labels)
ax.xaxis.grid(True)
ax.legend(loc='best')
ax.set_xlabel('Number of prizes')
# ...
```

❶ 建立水平直條圖，把 bar 換成 barh。

❷ 既然轉了個方向，那麼水平軸和垂直軸也要交換。

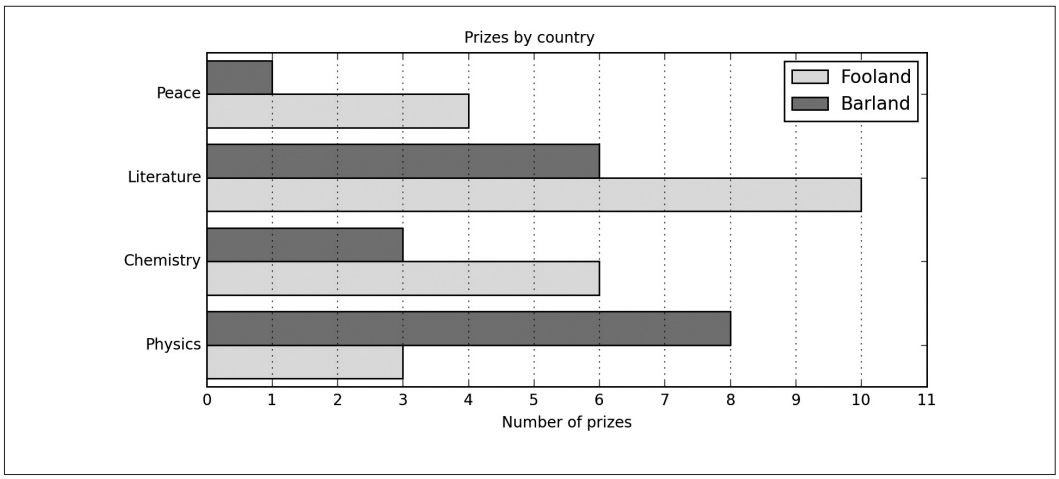


圖 10-7　改變直條方向

使用 Matplotlib 很容易就能畫出堆疊直條圖[9]，範例 10-8 把圖 10-6 轉成堆疊形式，結果請見圖 10-8。技巧在於 bar 的參數 bottom，把上直條的底部、設為下直條的頂部。

範例 10-8　把範例 10-6 轉成堆疊直條

```
# ...
bar_width = 0.8 # 直條寬度
xlocs = np.arange(len(foo_data))
ax.bar(xlocs, foo_data, bar_width, color='#fde0bc',  ❶
       label='Fooland')
ax.bar(xlocs, bar_data, bar_width, color='peru',     ❷
       label='Barland', bottom=foo_data)
# --- 標號、標籤、格狀、標題
```

9　關於堆疊直條圖是否為呈現群組資料的好方式，許多人存有疑問，請參考 Solomon Messing 的部落格（*http://bit.ly/1V2dG6B*），討論詳盡，而且有個「良好」的運用範例。

在範例 10-11 裡，使用 NumPy 非常好用的函式 polyfit，根據 x 與 y 陣列給定的點座標，產生出傾斜度和常數，然後在同一張圖表上頭繪製這條線。

範例 10-11　散布圖與迴歸線

```
num_points = 100
gradient = 0.5
x = np.array(range(num_points))
y = np.random.randn(num_points) * 10 + x*gradient
fig, ax = plt.subplots(figsize=(8, 4))
ax.scatter(x, y)
m, c = np.polyfit(x, y ,1)  ❶
ax.plot(x, m*x + c)  ❷
fig.suptitle('Scatterplot With Regression-line')
```

❶ 使用 NumPy 的 polyfit，參數 1 代表一維，算出直線傾斜度（m 和常數（c），代表穿過隨機點的最適切直線。

❷ 使用傾斜度和常數，在散布圖上繪製直線（y = mx + c）。

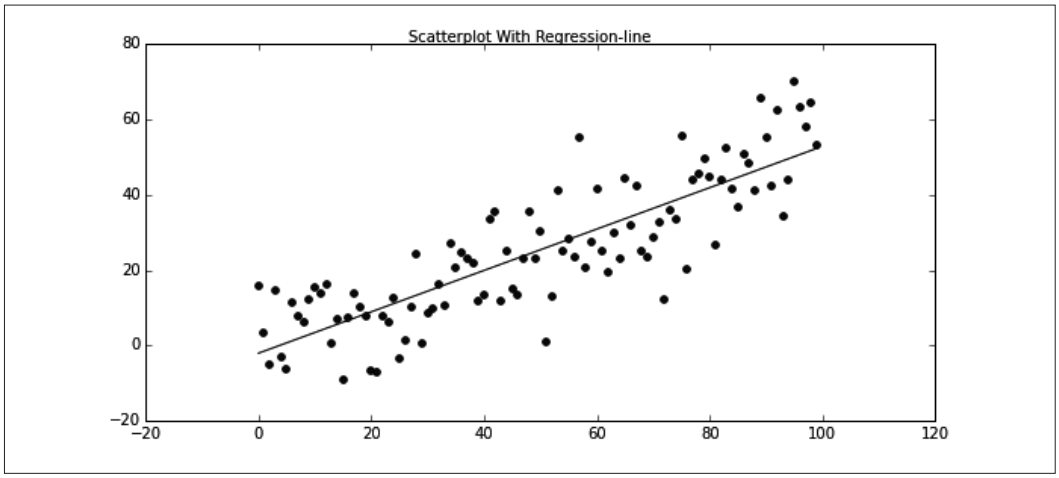


圖 10-11　散布圖與迴歸線

繪製迴歸線時，也應畫上信賴區間（confidence interval），一般來說這是個好主意，讓觀看者能夠根據點的數量與分佈情形，大概了解那條線有多可靠。使用 Matplotlib 與 NumPy 可求出信賴區間，但寫法有點兒笨拙，幸好，另有一套基於 Matplotlib 所打造的程式庫，特別為統計分析和資料視覺化工作提供額外的專門函式，而且許多人都認為，